REVERSE ENGINEERING

Tutorial for Node JS application using Sequelize and Passport,

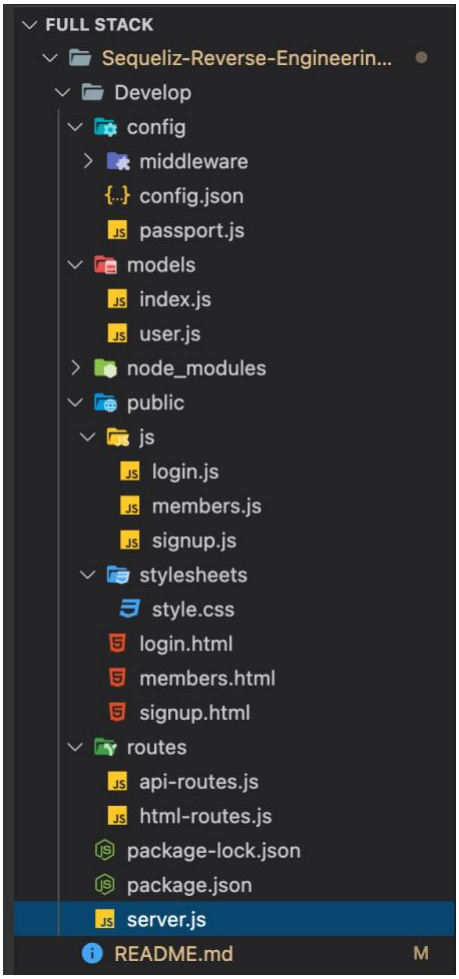
OVERVIEW & PURPOSE

This tutorial serves as a ‘walk-through’ for developers to familiarize themselves with a new codebase. This codebase can then be used to start a new project.

Intention of this Project (What does this codebase do)

This codebase is for password authentication. It allows a user to create an account, log into an account and sign back out securely - all on website files. All user data is stored in a MySQL Database.

FILE DIRECTORY MAP



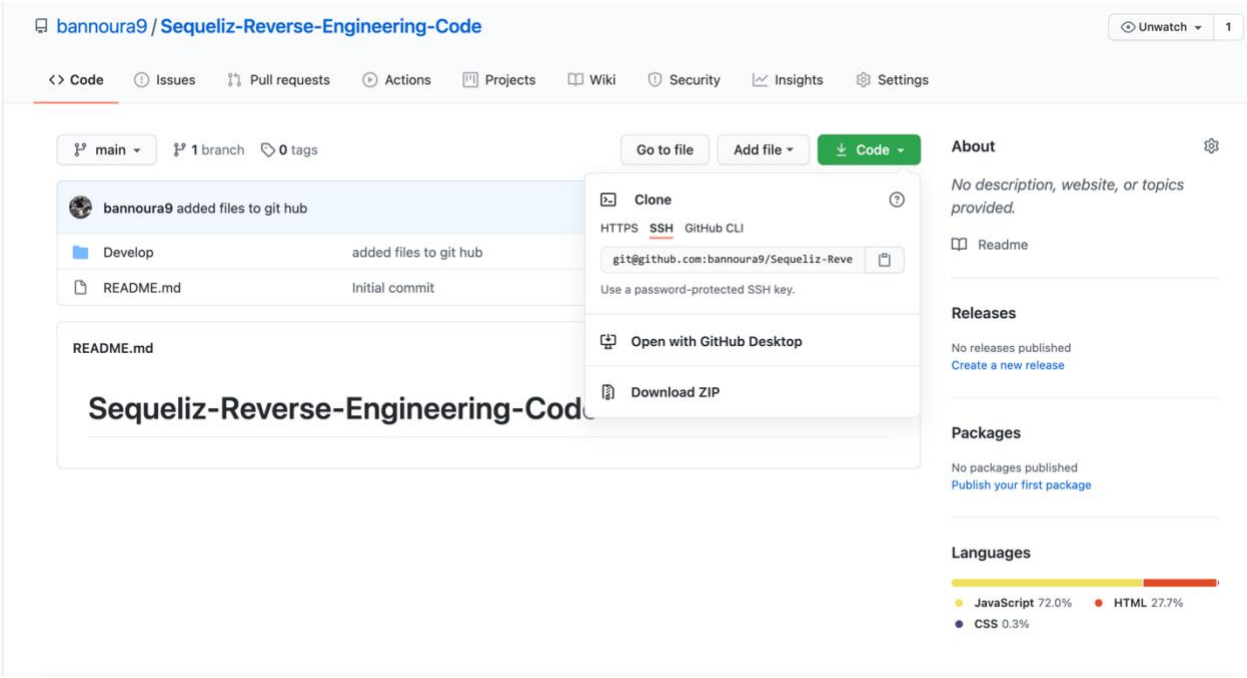
WALKTHROUGH OF CODE

Quick Instructions

1. Begin using this codebase by first cloning the repository into your local storage.

Example below, Gitlab repo used:

https://github.com/bannoura9/Sequeliz-Reverse-Engineering-Code



2. Once complete - change all const/lets in code to var to make the code run

3. Create SQL Scripts for databases as named in the config.json file passport_demo

4. Open and run scripts in MySQL to create the database.

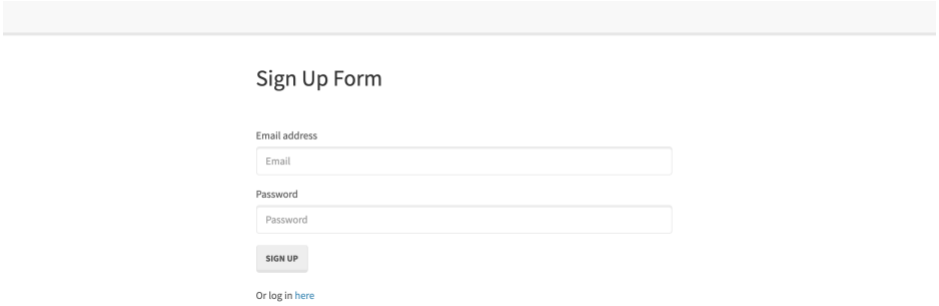
5. Edit the config.js file and include your own personal data (i.e.: password for MySQL).

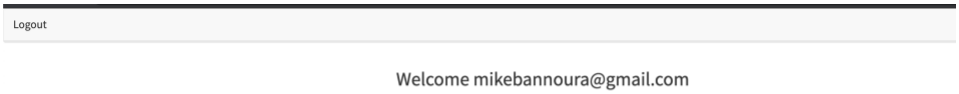
6. Open integrated terminal and run “NPM install” to install the required node modules for this project.

7. Run NPM audit fix if any issues

8. Run Node Server.JS to check server is working

9. Test html features (3 sites) by opening the local server link in your web browser.





Welcome mikebannoura@gmail.com

10. Check the mySQL database to ensure login data is saved.

File Walk Through with Steps

- Develop

This middleware file restricts routes that the user is not allowed to visit if they are not logged in. For example: the user cannot reach the members page if not logged in. If the user is logged in, it continues with request. This is the security feature of this passport codebase.

- config.json

middleware connection configuration to connect to server.

- passport.js

Middleware contains JavaScript logic that tells passport we want to log in with an email address and password.

models > index.js

This model connects to the database and imports each users log-in data.

- user.js

Requires "bcrypt" module for password hashing. This feature makes our database secure even if compromised. This javascript defines what is in our database.

- 📂 node_modules (+) (note: sub folders are not added in entirety in this tree as it would be too

- 📂 routes

📜 api-routes.js

Contains routes for signing in, logging out and getting users specific data to be displayed client on the client (browser) side.

- 📜 html-routes.js

Routes that check whether the user is signed in, whether user already has account etc and sends them to the correct html page.

- 📜 package.json

Contains all package info, node modules used, version info etc .

- 📜 passport_demo.sql
- 📜 server.js

Requires packages, sets up PORT, creates express and middleware, creates routes and syncs database / logs message in terminal on successful connection to server.

HOW TO IMPROVE CODEBASE

Describes what can be done next to start your project right.

A few tips on what you can do next to improve this code:

- Change all vars to consts and lets where relevant
- Add warnings when the user tries to sign up with a email that is already in the database
- Add warning when the password does not meet requirements
- Create some additional features in the members page to customize such as 'date joined' and edit password section.
- Potentially use AJAX to streamline API call functions
- Move isAuthenticated to within the html routes file
- Integrate the app into a dummy website to practically demonstrate its features.

  A relevant example might be a shopping site/ login to access Wishlist, shopping cart, checkout etc. Another example might be a forum.
- Create GitHub and Heroku repos to host your new project and REMEMBER tocreate a .gitignore file for your node modules!